

DOCKET No.

BVOC008

U.S. PATENT APPLICATION  
FOR  
SYSTEM, METHOD AND COMPUTER  
PROGRAM PRODUCT FOR A VOICE-ENABLED  
UNIVERSAL FLIGHT INFORMATION FINDER

ASSIGNEE: BEVOCAL, INC.

CARLTON FIELDS, LLP  
P.O. Box 721030  
SAN JOSE, CA 95172

# SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR A VOICE-ENABLED UNIVERSAL FLIGHT INFORMATION FINDER

## RELATED APPLICATIONS

5

The present application is related to a co-pending application which was filed concurrently herewith under the title "SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR A DISTRIBUTED SPEECH RECOGNITION TUNING PLATFORM" which is incorporated herein by reference in its entirety.

10

## FIELD OF THE INVENTION

The present invention relates to speech recognition, and more particularly to large-scale speech recognition.

15

## BACKGROUND OF THE INVENTION

Techniques for accomplishing automatic speech recognition (ASR) are well known. Among known ASR techniques are those that use grammars. A grammar is a representation of the language or phrases expected to be used or spoken in a given context. In one sense, then, ASR grammars typically constrain the speech recognizer to a vocabulary that is a subset of the universe of potentially-spoken words; and grammars may include subgrammars. An ASR grammar rule can then be used to represent the set of "phrases" or combinations of words from one or more grammars or subgrammars that may be expected in a given context. "Grammar" may also refer generally to a statistical language model (where a model represents phrases), such as those used in language understanding systems.

20

25

30

Products and services that utilize some form of automatic speech recognition ("ASR") methodology have been recently introduced commercially. For example,

AT&T has developed a grammar-based ASR engine called WATSON that enables development of complex ASR services. Desirable attributes of complex ASR services that would utilize such ASR technology include high accuracy in recognition; robustness to enable recognition where speakers have differing accents or dialects, and/or in the presence of background noise; ability to handle large vocabularies; and natural language understanding. In order to achieve these attributes for complex ASR services, ASR techniques and engines typically require computer-based systems having significant processing capability in order to achieve the desired speech recognition capability. In addition to WATSON, numerous ASR services are available which are typically based on personal computer (PC) technology.

One application of ASR techniques is the voice entry of addresses, i.e. street names, cities, etc. for the purpose of receiving directions. One example of such application is disclosed in U.S. Patent Number 6,108,631. Such invention relates to an input system for at least location and/or street names, including an input device, a data source arrangement which contains at least one list of locations and/or streets, and a control device which is arranged to search location or street names, entered via the input device, in a list of locations or streets in the data source arrangement. In order to simplify the input of location and/or street names, the data source arrangement contains not only a first list of locations and/or streets with alphabetically sorted location and/or street names, but also a second list of locations and/or streets with location and/or street names sorted on the basis of a frequency criterion. A speech input system of the input device conducts input in the form of speech to the control device. The control device is arranged to perform a sequential search for a location or street name, entered in the form of speech, as from the beginning of the second list of locations and/or streets.

Such prior art direction services supply to a traveler automatically developed step-by-step directions for travel from a starting point to a destination. Typically these directions are a series of steps which detail, for the entire route, a) the particular

series of streets or highways to be traveled, b) the nature and location of the entrances and exits to/from the streets and highways, e.g., turns to be made and exits to be taken, and c) optionally, travel distances and landmarks.

- 5 There is therefore a need for additional applications of such technology.

## DISCLOSURE OF THE INVENTION

A system, method and computer program product are afforded for providing voice-enabled driving directions. Initially, an utterance is received representative of a  
5 flight identifier. Utilizing a speech recognition process, the utterance is then transcribed. Further, a database is queried for generating flight information based on the flight identifier.

In one embodiment of the present invention, the utterance may be received utilizing  
10 a network, i.e. the Internet. Further, the flight information may include a time of arrival of the flight. Still yet, the flight identifier may include a flight number.

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
22

## **BRIEF DESCRIPTION OF THE DRAWINGS**

Figure 1 illustrates an exemplary environment in which the present invention may be  
5 implemented;

Figure 2 shows a representative hardware environment associated with the computer  
systems of Figure 1;

10 Figure 3 is a schematic diagram showing one exemplary combination of databases  
that may be used for generating a collection of grammars;

Figure 4 illustrates a gathering method for collecting a large number of grammars  
such as all of the street names in the United States of America using the combination  
15 of databases shown in Figure 3;

Figure 4A illustrates a pair of exemplary lists showing a plurality of streets names  
organized according to city;

20 Figure 5 illustrates a plurality of databases of varying types on which the grammars  
may be stored for retrieval during speech recognition;

Figure 6 illustrates a method for speech recognition using heterogeneous protocols  
associated with the databases of Figure 5;

25 Figure 7 illustrates a method for providing a speech recognition method that  
improves the recognition of street names, in accordance with one embodiment; and

Figures 8-11 illustrate an exemplary speech recognition process, in accordance with  
30 one embodiment of the present invention;



## DESCRIPTION OF THE PREFERRED EMBODIMENTS

- Figure 1 illustrates an exemplary environment **100** in which the present invention may be implemented. As shown, a plurality of computers **102** are interconnected via a network **104**. In one embodiment, such network includes the Internet. It should be noted, however, that any type of network may be employed, i.e. local area network (LAN), wide area network (WAN), etc.
- Figure 2 shows a representative hardware environment associated with the computer systems **102** of Figure 1. Such figure illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **210**, such as a microprocessor, and a number of other units interconnected via a system bus **212**.
- The workstation shown in Figure 2 includes a Random Access Memory (RAM) **214**, Read Only Memory (ROM) **216**, an I/O adapter **218** for connecting peripheral devices such as disk storage units **220** to the bus **212**, a user interface adapter **222** for connecting a keyboard **224**, a mouse **226**, a speaker **228**, a microphone **232**, and/or other user interface devices such as a touch screen (not shown) to the bus **212**, communication adapter **234** for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter **236** for connecting the bus **212** to a display device **238**. The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned.



- A preferred embodiment is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions require adaptation to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided.
- 10 OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not
- 15 require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.
- 20 In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a component integration architecture. A component integration architecture is a set of architecture mechanisms which allow software modules in different
- 25 process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture. It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can
- 30 be formed.

OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

OOP also allows creation of an object that “depends from” another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further limitation or detail to it. The object representing the ceramic piston engine “depends from” the object representing the piston engine. The relationship between these objects is called inheritance.

When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function

behind the same name is called polymorphism and it greatly simplifies communication among objects.

5 With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, one's logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

- 10 • Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.
- Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.
- 15 • An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.
- An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

20 With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is a model of some aspects of reality, whether that reality is a physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

25 If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from

which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built objects.

This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, Common Lisp Object System (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

- Objects and their corresponding classes break down complex programming problems into many smaller, simpler problems.
- Encapsulation enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from accidental damage, but allows other objects to interact with that data by calling the object's member functions and structures.
- Subclassing and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in

the system. Thus, new capabilities are created without having to start from scratch.

- Polymorphism and multiple inheritance make it possible for different programmers to mix and match characteristics of many different classes and create specialized objects that can still work with related objects in predictable ways.
- Class hierarchies and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.
- Libraries of reusable classes are useful in many situations, but they also have some limitations. For example:
  - Complexity. In a complex system, the class hierarchies for related classes can become extremely confusing, with many dozens or even hundreds of classes.
  - Flow of control. A program written with the aid of class libraries is still responsible for the flow of control (i.e., it must control the interactions among all the objects created from a particular library). The programmer has to decide which functions to call at what times for which kinds of objects.
  - Duplication of effort. Although class libraries allow programmers to use and reuse many small pieces of code, each programmer puts those pieces together in a different way. Two different programmers can use the same set of class libraries to write two programs that do exactly the same thing but whose internal structure (i.e., design) may be quite different, depending on hundreds of small decisions each programmer makes along the way. Inevitably, similar pieces of code end up doing similar things in slightly different ways and do not work as well together as they should.

Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over again. A relatively new extension of the class library concept is to have a framework of class libraries. This framework is more complex and consists of significant

collections of collaborating classes that capture both the small-scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer's code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing control in this way to users, the developer creates a program that is much easier to use.

Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after it's called by the event loop. Application code still "sits on top of" the system.

Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.

- 10 Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer's code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

- A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

- 25 Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.

There are three main differences between frameworks and class libraries:

- Behavior versus protocol. Class libraries are essentially collections of behaviors that you can call when you want those individual behaviors in your program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.
- Call versus override. With a class library, the code the programmer instantiates objects and calls their member functions. It's possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework's reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces of software that are called by the framework rather than specifying how the different pieces should work together.
- Implementation versus design. With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

25

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the Newco. HTTP or other protocols could be readily

30



substituted for HTML without undue experimentation. Information on these products is available in T. Berners-Lee, D. Connolly, "RFC 1866: Hypertext Markup Language - 2.0" (Nov. 1995); and R. Fielding, H. Frystyk, T. Berners-Lee, J. Gettys and J.C. Mogul, "Hypertext Transfer Protocol -- HTTP/1.1: HTTP Working Group  
5 Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an  
10 application of ISO Standard 8879; 1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing  
15 computing resources. Until recently, HTML has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

- Poor performance;
- Restricted user interface capabilities;
- 20 • Can only produce static Web pages;
- Lack of interoperability with existing applications and data; and
- Inability to scale.

Sun Microsystem's Java language solves many of the client-side problems by:

- 25 • Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom  
30 "widgets" (e.g., real-time stock tickers, animated icons, etc.) can be created, and

client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

5

- Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets." Java applets are small, specialized applications that comply with Sun's Java Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g., simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g., Netscape Navigator) by copying code from the server to client. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically, "C++ with extensions from Objective C for more dynamic method resolution."
- 20 Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server
- 30

applications. One of ordinary skill in the art readily recognizes that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.

### *Preferred Embodiments*

5

Initially, a database must first be established with all of the necessary grammars. In one embodiment of the present invention, the database is populated with a multiplicity of street names for voice recognition purposes. In order to get the best coverage for all the street names, data from multiple data sources may be merged.

10

Figure 3 is a schematic diagram showing one exemplary combination of databases **300**. In the present embodiment, such databases may include a first database **302** including city names and associated zip codes (i.e. a ZIPUSA database), a second database **304** including street names and zip codes (i.e. a Geographic Data Technology (GDT) database), and/or a United States Postal Services (USPS) database **306**. In other embodiments, any other desired databases may be utilized. Further tools may also be utilized such as a server **308** capable of verifying street, city names, and zip codes.

15

20

Figure 4 illustrates a gathering method **400** for collecting a large number of grammars such as all of the street names in the United States of America using the combination of databases **300** shown in Figure 3. As shown in Figure 4, city names and associated zip code ranges are initially extracted from the ZIPUSA database. Note operation **402**. It is well known in the art that each city has a range of zip codes associated therewith. As an option, each city may further be identified using a state and/or county identifier. This may be necessary in the case where multiple cities exist with similar names.

25

30

Next, in operation **404**, the city names are validated using a server capable of verifying street names, city names, and zip codes. In one embodiment, such server

may take the form of a MapQuest server. This step is optional for ensuring the integrity of the data.

Thereafter, all of the street names in the zip code range are extracted from USPS data in operation 406. In a parallel process, the street names in the zip code range are similarly extracted from the GDT database. Note operation 408. Such street names are then organized in lists according to city. Figure 4A illustrates a pair of exemplary lists 450 showing a plurality of streets names 452 organized according to city 454. Again, in operation 410, the street names are validated using the server capable of verifying street names, city names, and zip codes.

It should be noted that many of the databases set forth hereinabove utilize abbreviations. In operation 412, the street names are run through a name normalizer, which expands common abbreviations and digit strings. For example, the abbreviations "St." and "Cr." can be expanded to "street" and "circle," respectively.

In operation 414, a file is generated for each city. Each of such files delineates each of the appropriate street names.

Figure 5 illustrates a plurality of databases 500 of varying types on which the grammars may be stored for retrieval during speech recognition. The present embodiment takes into account that only a small portion of the grammars will be used heavily used during use. Further, the overall amount of grammars is so large that it is beneficial for it to be distributed across several databases. Because network connectivity is involved, the present embodiment also provides for a fail-over scheme.

As shown in Figure 5, a plurality of databases 500 are included having different types. For example, such databases may include a static database 504, dynamic database 506, web-server 508, file system 510, or any other type of database. Table 1 illustrates a comparison of the foregoing types of databases.

Table 1

	When Compiled	On Server?	Protocol
Static	Offline	Yes	Proprietary Vendor
Dynamic	Offline	No	ORACLE™ OCI
Web server	Runtime	No	HTTP
File System	Runtime	No	File System Access

5     Figure 6 illustrates a method 600 for speech recognition using heterogeneous protocols associated with the databases of Figure 5. Initially, in operation 602, a plurality of grammars, i.e. street names, are maintained in databases of different types. In one embodiment, the types may include static, dynamic, web server, and/or file system, as set forth hereinabove.

10

During use, in operation 604, the grammars are dynamically retrieved utilizing protocols based on the type of the database. Retrieval of the grammars may be initially attempted from a first database. The database subject to such initial attempt may be selected based on the type, the specific content thereof, or a combination thereof.

15

For example, static databases may first be queried for the grammars to take advantage of their increased efficiency and speed, while the remaining types may be used as a fail-over mechanism. Moreover, the static database to be initially queried may be populated with grammars that are most prevalently used. By way of example, a static database with just New York streets may be queried in response to a request from New York. As such, one can choose to include certain highly used grammars as static grammars (thus reducing network traffic), while other databases with lesser used grammars may be accessible through various other network protocols.

20

25

Further, by storing the same grammar in more than one node in such a distributed architecture, a control flow of the grammar search algorithm could point to a redundant storage area if required. As such, a fail-over mechanism is provided. By way of example, in operation 606, it may be determined whether the grammars may be retrieved from a first one of the databases during a first attempt. Upon the failure of the first attempt, the grammars may be retrieved from a second one of the databases, and so on. Note operation 608.

10 The present approach thus includes distributing grammar resources across a variety of data storage types (static packages, dynamic grammar databases, web servers, file systems), and allows the control flow of the application to search for the grammars in all the available resources until it is found.

15 Figure 7 illustrates a method 700 for providing a speech recognition method that improves the recognition of street names, in accordance with one embodiment of the present invention. In order to reduce the phonetic confusability due to the existence of smaller streets whose names happen to be phonetically similar to that of more popular streets, traffic count statistics may be used when recognizing the grammars  
20 to weigh each street.

During operation 702, a database of words is maintained. Initially, in operation 704, a probability is assigned to each of the words, i.e. street names, which indicates a prevalence of use of the word. As an option, the probability may be determined  
25 using statistical data corresponding to use of the streets. Such statistical data may include traffic counts such as traffic along the streets and along intersecting streets.

The traffic count information may be given per intersection. One proposed scheme to extract probabilities on a street-to-street basis will now be set forth. The goal is to  
30 include in the grammar probabilities for each street that would predict the likelihood

users will refer to it. It should be noted that traffic counts are an empirical indication of the importance of a street.

5 In use, data may be used which indicates an amount of traffic at intersections of streets. Equation #1 illustrates the form of such data. It should be noted that data in such form is commonly available for billboard advertising purposes.

#### Equation #1

10       TrafficIntersection(streetA, streetB) = X  
          TrafficIntersection(streetA, streetC) = Y  
          TrafficIntersection(streetA, streetD) = Z  
          TrafficIntersection(streetB, streetC) = A

15 To generate a value corresponding to a specific street, all of the intersection data involving such street may be aggregated. Equation #2 illustrates the manner in which the intersection data is aggregated for a specific street.

#### Equation #2

20                               Traffic(streetA) = X + Y + Z

25 The aggregation for each street may then be normalized. One exemplary method of normalization is represented by Equation #3.

#### Equation #3

30                               Normalization [Traffic(streetA)] =  $\log_{10}(X + Y + Z)$

Such normalized values may then be used to categorize each of the streets in terms of prevalence of use. Preferably, this is done separately for each city. Each category

is assigned a constant scalar associated with the popularity of the street. By way of example, the constant scalars 1, 2 and 3 may be assigned to normalized aggregations .01, .001, and .0001, respectively. Such popularity may then be added to the city grammar file to be used during the speech recognition process.

5

During use, an utterance is received for speech recognition purposes. Note operation 706. Such utterance is matched with one of the words in the database based at least in part on the probability, as indicated by operation 708. For example, when confusion is raised as to which of two or more streets an utterance is referring, the street with the highest popularity (per the constant scalar indicator) is selected as a match.

10

### *Exemplary Speech Recognition Process*

15 An exemplary speech recognition process will now be set forth. It should be understood that the present example is offered for illustrative purposes only, and should not be construed as limiting in any manner.

Figure 8 shows a timing diagram which represents the voice signals in A. According to the usual speech recognition techniques, such as explained in above-mentioned European patent, evolutionary spectrums are determined for these voice signals for a time tau represented in B in Figure 8 by the spectral lines R1, R2 . . . . The various lines of this spectrum obtained by fast Fourier transform, for example, constitute vectors. For determining the recognition of a word, these various lines are compared with those established previously which form the dictionary and are stored in memory.

25

Figure 9 shows the flow chart which explains the method according to the invention. Box K0 represents the activation of speech recognition; this may be made by validating an item on a menu which appears on the screen of the device. Box K1 represents the step of the evaluation of ambient noise. This step is executed between

30



the instants  $t_0$  and  $t_1$  (see Figure 8) between which the speaker is supposed not to speak, i.e. before the speaker has spoken the word to be recognized. Supposing  $N_b$  is this value which is expressed in dB relative to the maximum level (if one works with 8 bits, this maximum level 0 dB is given by 1111 1111). This measure is taken considering the mean value of the noise vectors, their moduli, or their squares. From this level measured in this manner is derived a threshold  $TH$  (box K2) as a function of the curve shown in Figure 10.

Box K2a represents the breakdown of a spoken word to be recognized into input vectors  $V_i$ . Box K3 indicates the computation of the distances  $d^k$  between the input vectors  $V_i$  and the reference vectors  $w_i^k$ . This distance is evaluated based on the absolute value of the differences between the components of these vectors. In box K4 is determined the minimum distance  $D^B$  among the minimum distances which have been computed. This minimum value is compared with the threshold value  $TH$ , box K5. If this value is higher than the threshold  $TH$ , the word is rejected in box K6, if not, it is declared recognized in box K7.

The order of various steps may be reversed in the method according to the invention. As this is shown in Figure 11, the evaluation of the ambient noise may also be carried out after the speaker has spoken the word to be recognized, that is, between the instants  $t_0'$  and  $t_1'$  (see Figure 8). This is translated in the flow chart of Figure 11 by the fact that the steps K1 and K2 occur after step K4 and before decision step K5.

The end of this ambient noise evaluation step, according to a characteristic feature of the invention, may be signaled to the speaker in that a beep is emitted, for example, by a loudspeaker which then invites the speaker to speak. The present embodiment has taken into account that a substantially linear function of the threshold value as a function of the measured noise level in dB was satisfactory. Other functions may be found too, without leaving the scope of the invention therefore.

If the distances vary between a value from 0 to 100, the values of TH1 may be 10 and those of TH2 80 for noise levels varying from -25 dB to -5 dB.

### *Exemplary Applications*

5

Various applications of the foregoing technology will now be set forth. It should be noted that such applications are for illustrative purposes, and should not be construed limiting in any manner.

10 Figure 12 illustrates a method 1200 for providing voice-enabled driving directions. Initially, in operation 1202, an utterance representative of a destination address is received. It should be noted that the addresses may include street names or the like. Such utterance may also be received via a network.

15 Thereafter, in operation 1204, the utterance is transcribed utilizing a speech recognition process. As an option, the speech recognition process may include querying one of a plurality of databases based on the origin address. Such database that is queried by the speech recognition process may include grammars representative of addresses local to the origin address.

20

An origin address is then determined. Note operation 1206. In one embodiment of the present invention, the origin address may also be determined utilizing the speech recognition process. It should be noted that global positioning system (GPS) technology or other methods may also be utilized for such purpose.

25

A database is subsequently for queried generating driving directions based on the destination address and the origin address, as indicated in operation 1208. In particular, a server (such as a MapQuest server) may be utilized to generate such driving directions. Further, such driving directions may optionally be sounded out

30

via a speaker or the like.

Figure 13 illustrates a method 1300 for providing voice-enabled driving directions based on a destination name. Initially, in operation 1302, an utterance representative of a destination name is received. Optionally, the destination name may include a category and/or a brand name. Such utterance may be received via a network.

In response to the receipt thereof, the utterance is transcribed utilizing a speech recognition process. See operation 1304. Further, in operation 1306, a destination address is identified based on the destination name. It should be noted that the addresses may include street names. To accomplish this, a database may be utilized which includes addresses associated with business names, brand names, and/or goods and services. Optionally, such database may include a categorization of the goods and services, i.e. virtual yellow pages, etc.

Still yet, an origin address is identified. See operation 1308. In one embodiment of the present invention, the origin address may be determined utilizing the speech recognition process. It should be noted that global positioning system (GPS) technology or other techniques may also be utilized for such purpose.

Based on such destination name and origin address, a database is subsequently queried for generating driving directions. Note operation 1310. Similar to the previous embodiment, a server (such as a MapQuest server) may be utilized to generate such driving directions, and such driving directions may optionally be sounded out via a speaker or the like.

Figure 14 illustrates a method 1400 for providing voice-enabled driving directions. Initially, in operation 1402, an utterance is received representative of a flight identifier. Optionally, the flight identifier may include a flight number. Further, such utterance may be received via a network.

Utilizing a speech recognition process, the utterance is then transcribed. Note operation **1404**. Further, in operation **1406**, a database is queried for generating flight information based on the flight identifier. As an option, the flight information may include a time of arrival of the flight, a flight delay, or any other information  
5 regarding a particular flight.

Figure **15** illustrates a method **1500** for providing localized content. Initially, an utterance representative of content is received from a user. Such utterance may be received via a network. Note operation **1502**. In operation **1504**, such utterance is  
10 transcribed utilizing a speech recognition process.

A current location of the user is subsequently determined, as set forth in operation **1506**. In one embodiment of the present invention, the current location may be determined utilizing the speech recognition process. In another embodiment of the  
15 present invention, the current location may be determined by a source of the utterance. This may be accomplished using GPS technology, identifying a location of an associated inputting computer, etc.

Based on the transcribed utterance and the current location, a database is queried for  
20 generating the content. See operation **1508**. Such content may, in one embodiment, include web-content taking the form of web-pages, etc.

As an option, the speech recognition process may include querying one of a plurality of databases based on the current address. It should be noted that the database  
25 queried by the speech recognition process may include grammars representative of the current location, thus facilitating the retrieval of appropriate content.

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the  
30 breadth and scope of a preferred embodiment should not be limited by any of the

